NP-трудност и **NP**-пълнота

Тодор Дуков

$\mathbf{P} = \mathbf{NP}$ или $\mathbf{P} \neq \mathbf{NP}$? Колко пък да е трудно?

За съжаление все още няма отговор на този въпрос – не се знае дали $\mathbf{P} = \mathbf{NP}$ или $\mathbf{P} \neq \mathbf{NP}$. Най-доброто, което имаме до момента, са няколко задачи в класа **NP**, които са изключително важни. Тези задачи в някакъв смисъл характеризират целия клас. Тях ще ги наричаме **NP**-пълни. Формалната дефиниция е следната, една изчислителна задача $\mathbf{X}$ е **NP**-пълна, ако:

- $\mathbf{X}$ е в класа **NP**;
- всяка задача $\mathbf{Y}$ в класа **NP** може алгоритмично да се сведе до задачата $\mathbf{X}$ за полиномиално време.

Фактът, че $\mathbf{Y}$ се свежда до $\mathbf{X}$ за полиномиално време ще бележим с $\mathbf{Y} \leq_p \mathbf{X}$*. Лесно се вижда, че $\leq_p$ е транзитивна. Когато второто условие е изпълнено (без да е непременно изпълнено първото) казваме, че задачата $\mathbf{X}$ е **NP**-трудна. Интересното за **NP**-пълните задачи е следното:

- ако покажем, че която и да е задача в **NP** се решава за полиномиално време, то тогава $\mathbf{P} = \mathbf{NP}$;
- ако покажем, че която и да е задача в **NP** не може да се реши за полиномиално време, то тогава $\mathbf{P} \neq \mathbf{NP}$.

Всички **NP**-пълни задачи са еквивалентни в смисъл, че всяка може да се сведе до другата за полиномиално време. Така имайки полиномиално алгоритъм, който решава някоя **NP**-пълна задача $\mathbf{X}$, то тогава можем да решим всяка задача $\mathbf{Y}$ от **NP** за полиномиално време:

```
1 РешиY(ВходY):
2     ВходX ← РедуцирайВходYДоВходXЗаПолиномиалноВреме(ВходY)
3     ИзходX ← РешиXЗаПолиномиалноВреме(ВходX)
4     ИзходY ← РедуцирайИзходXДоИзходYЗаПолиномиалноВреме(ИзходX)
5     върни ИзходY
```

Обаче ние нито имаме такъв алгоритъм, нито не знаем, че такъв алгоритъм няма. Това са задачи, които хем не можем да решим ефективно, хем не можем да покажем, че такова решение няма. Най-доброто, което можем да направим, е да покажем, задачата е еквивалентна по трудност на други задачи, за които други много умни хора не са се сетили.

"Основната" **NP**-пълна задача

По-принцип е трудно директно да се показва **NP**-пълнота, ако се кара по дефиницията. Това, което обикновено се прави, е се показва по дефиниция, че една задача е **NP**-пълна (все трябва да започнем от някъде), след което се правят полиномиални редукции от задачи, за които знаем, че са **NP**-пълни, към задачите, която искаме да покажем, че са **NP**-пълни. Тук се възползваме от транзитивността на $\leq_p$. Разбира се, това само би показало **NP**-трудност, трябва и да се провери принадлежност към класа **NP**. Задачата, от която обикновено се почва е тази за удовлетворимост/изпълнимост.

Теорема (Кук-Левин)**.** *Задачата **SAT** е **NP**-пълна.*

След това се показва, че **3SAT** $\leq_p$ **SAT**. Ние вече знаем, че тя е в класа **NP**, така че ако направим тази редукция, ще излезе, че **3SAT** е **NP**-пълна задача. Можем да направим следната редукция – за всеки дизюнкт D във входната формула φ:

- ако в D участва точно един литерал L, то тогава избираме нови променливи x и y, и заменяме D с конюнкцията на дизюнктите $(L \vee x \vee y), (L \vee x \vee \overline{y}), (L \vee \overline{x} \vee y)$ и $(L \vee \overline{x} \vee \overline{y})$;
- ако в D участват два литерала L_1, L_2, то тогава избираме нова променлива x, и заменяме D с конюнкцията на дизюнктите $(L_1 \vee L_2 \vee x)$ и $(L_1 \vee L_2 \vee \overline{x})$;

*На други места вместо $\leq_p$ се използват означенията $\leq_m^p$ и $\propto_p$.

- ако в D участват три литерала, не променяме D;
- ако в D участват литералите $L_1, \ldots, L_n$, където $n > 3$, то тогава избираме нови променливи $x_1, \ldots, x_{n-3}$, и заменяме D с конюнкцията на дизюнктите

$$(L_1 \vee L_2 \vee x_1), (L_3 \vee \overline{x_1} \vee x_2), (L_4 \vee \overline{x_2} \vee x_3), \ldots, (L_{n-2} \vee \overline{x_{n-4}} \vee x_{n-3}), (L_{n-1} \vee L_n \vee \overline{x_{n-3}}).$$

Накрая ще получим като резултат формула ψ, която не е еквивалентна на φ, но е изпълнима т.с.т.к. φ е изпълнима. На читателя оставяме да провери, че това е вярно. Остана само да проверим, че всичко това става за полиномиално време. Дължината на ψ ще бъде полиномиална спрямо тази на φ, защото:

- дизюнктите с един литерал се заменят с четири дизюнкта с три литерала;
- дизюнктите с два литерала се заменят с два дизюнкта с три литерала;
- дизюнктите с три литерала не се променят;
- дизюнктите с $n > 4$ литерала се заменят с $n - 2$ дизюнкта с три литерала.

По-съмнителното е търсенето на нови променливи, но и това няма как да е прекалено бавно, защото броят на променливите, които участват в една формула е по-малък или равен на дължината ѝ. Тъй като ние ще получим формула с полиномиална на φ дължина, в нея ще участват най-много полиномиално на φ променливи. Това означава, че ако всеки път търсим неизползваната променлива с най-малък индекс, няма да търсим прекалено дълго. С това получаваме, че **3SAT** е **NP**-трудна задача, и понеже сме показвали че е в класа **NP**, то тя е и **NP**-пълна.

Класически NP-пълни задачи

Ще покажем няколко класически примери за **NP**-пълни задачи. Тъй като за тях сме доказвали, че са в класа **NP**, ни остава само да покажем, че са **NP**-трудни. Нека сега покажем, че **3SAT** $\leq_p$ **Clique**. Нека φ е формула в 3КНФ със n на брой дизюнкта. За полиномиално време ще построим граф G, за които φ е изпълнима т.с.т.к. G съдържа n-клика. За всеки дизюнкт $(L_1(x) \vee L_2(y) \vee L_3(z))$, който участва във φ, в графа G има върховете от вида $\{\langle x, v_x \rangle, \langle y, v_y \rangle, \langle z, v_z \rangle\}$, където $v_x, v_y, v_z \in \{\mathbb{T}, \mathbb{F}\}$ и интерпретирайки t като v_t за $t \in \{x, y, z\}$, дизюнкта $(L_1(x) \vee L_2(y) \vee L_3(z))$ се оценява като $\mathbb{T}$. Ребра ще има между тези множества, които не си противоречат т.е. няма променлива x, за която $\langle x, \mathbb{T} \rangle$ да участва в едното множество и $\langle x, \mathbb{F} \rangle$ да участва в другото. По построение е очевидно, че в G има n клика т.с.т.к. φ е изпълнима. В едната посока кликата ни казва точно как да оценим променливите, а в другата посока от оценката можем да извлечем кликата. Тъй като за всеки дизюнкт получаваме най-много 2^3 върхове, то тогава конструкцията е полиномиална. С това показахме, че **Clique** е **NP**-трудна задача, откъдето е и **NP**-пълна.
Сега нека да видим, че **Clique** $\leq_p$ **VertexCover**. За входния граф $G = \langle V, E \rangle$ строим $\overline{G} = \langle V, \overline{E} \rangle$, където:

$$\overline{E} = \{(u, v) \mid u, v \in V \,\&\, (u, v) \notin E\}.$$

Този граф очевидно се строи за време $\Theta(|V|^2)$. Оказва се, че за всяко $X \subseteq V$ е изпълнено, че:

$$X \text{ е клика в } G \iff V \setminus X \text{ е върхово покритие в } \overline{G}.$$

($\Rightarrow$) Нека X е клика в G. Тогава което и ребро $(u, v) \in \overline{E}$ да вземем, $u \notin X$ или $v \notin X$. Ако $u, v \in X$, то тогава тъй като $(u, v) \in \overline{E}$, то тогава $(u, v) \notin E$, което противоречи с факта, че X е клика. Така наистина $V \setminus X$ е върхово покритие в $\overline{G}$.
($\Leftarrow$) Нека X не е клика в G. Тогава има два върха $u, v \in X$, за които $(u, v) \notin E$. Но тогава $(u, v) \in \overline{E}$, откъдето $V \setminus X$ не е върхово покритие в $\overline{G}$.
Псевдокода на редукцията би изглеждал така:

```
1  РешиCliqueЧрезVertexCover(G = (V, E) граф, k - естествено число):
2      инициализирай граф Ḡ = (V, ∅)
3
4      за всеки връх u ∈ V:
5          за всеки връх v ∈ V:
6              ако u ≠ v и (u, v) ∉ E:
7                  добави реброто (u, v) към Ḡ
8
9      върни РешиVertexCover(Ḡ, |V| − k)
```

С това показахме, че **VertexCover** е **NP**-трудна задача, откъдето е и **NP**-пълна.

Да видим, и че **Clique** $\leq_p$ **Anticlique**. Използвайки същата конструкция от предната задача, лесно се вижда, че:

$$X \text{ е клика в } G \iff X \text{ е антиклика в } \overline{G}$$

Псевдокода на редукцията би изглеждал така:

```
РешиCliqueЧрезAnticlique(G = (V, E) граф, k - естествено число):
    инициализирай граф Ḡ = (V, ∅)

    за всеки връх u ∈ V:
        за всеки връх v ∈ V:
            ако u ≠ v и (u, v) ∉ E:
                добави реброто (u, v) към Ḡ

    върни РешиAnticlique(Ḡ, k)
```

С това показахме, че **Anticlique** е **NP**-трудна задача. Принадлежността към класа **NP** е очевидна. Така задачата е **NP**-пълна.